Software Design

April 26, 2013

1. Introduction

1.1.1. Purpose of This Document

This document provides a high level description of the design and implementation of Cypress, an open source certification testing tool for the calculation of Clinical Quality Measures (CQMs) as part of the stage 2 Meaningful Use (MU2) requirements of the EHR Incentive Program.

This document has been written for software engineers, software architects, and technical program managers. The latest information about Cypress is available at the project's open source website[1].

1.1.2. Overview

Cypress is an open source (see section 5) certification testing tool that automates testing of the ability of electronic health record (EHR) systems to correctly calculate MU Stage 2 CQM results.

Cypress is recognized by the Office of the National Coordinator of Health IT (ONC) as the official test tool for use by Authorized Testing Laboratories (ATL) in CQM certification. In addition, Cypress can be used, free of charge, for pre-certification testing by any EHR developer.

The CQM certification criteria addressed by Cypress are:

- **Capture and Export** - the ability of an EHR system to capture the data required for each CQM for which the EHR system is being certified through its clinical user interface and to create a QRDA-I formatted data file with appropriate content that can be input by another EHR system for use in CQM calculation
- **Import and Calculate** - the ability of an EHR system to incorporate data (from other EHR systems where necessary) in QRDA-I format and correctly calculate the CQM results and present them in QRDA-III format
- **Electronic Submission** - the capability of an EHR system to create a standard data file that can be electronically accepted by the Centers for Medicaid & Medicare Services (CMS)

Cypress includes a standard set of synthetic patient records that exercise all of the MU Stage 2 CQMs. The same test patients are also used to support 'Capture and Export' and 'Import and Calculate' tests. Cypress test data is available in standard formats (QRDA category I and HTML) for both automated and manual input into an EHR system.

1.1.3. Concept of Operations

Figure 1 shows the high-level ATL workflow.

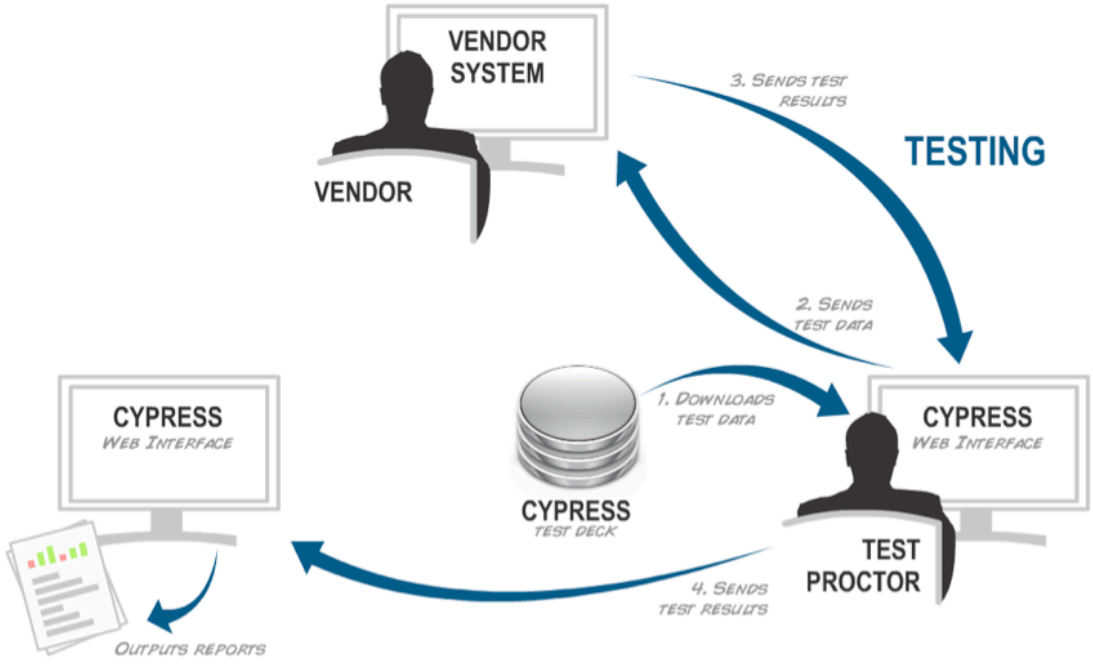

Figure 1: Authorized Testing Lab Workflow

The test proctor uses the Cypress user interface, a Web application, to:

- Register systems for test (not shown)
- Create test data to be entered into a system under test (2)
- Import test results from the system under test (3)
- Validate and report on test results (4)

The Cypress measure bundle (see section 2.1) is a separate download that must be manually installed prior to use of the Cypress application. The bundle contains a particular released version of the CQMs and the associated version of the required test patient data.

In step (2), test data is exported by Cypress as either QRDA category I or HTML. QRDA category I is suitable for automated import into an EHR system under test (for 'import and calculate' only). HTML is suitable for manual entry of data for 'capture and export'.

In step (3), test results are required to be formatted as QRDA category I for "capture and export" testing and as QRDA category III for "import and calculate" testing.

In step (4) the Cypress application presents the results of testing to the test proctor. The application highlights errors in both the vendor-generated results and in the structure of vendor-generated QRDA documents.

2. Web Application

Cypress is a Ruby on Rails 3 Web application; the source code can be downloaded from the project GitHub repository[2]. Cypress uses the open source MongoDB document-based database for data persistence. Cypress relies on a number of other standard and custom-developed modules to provide its functionality and uses Bundler to manage these dependencies. The custom-developed modules that Cypress relies on are:

- **Quality Measure Engine**
  Manages the calculation of CQM results, see section 3.4.
- **Health Data Standards**
  Provides import and export capability for a variety of clinical data formats including HQMF and QRDA (categories I and III). Data is stored in MongoDB as JSON documents. Ruby models, using the Mongoid library, define the format of those JSON documents. See sections 3.1 and 3.4.
- **HQMF2JS**
  Provides a translation from HQMF to JavaScript suitable for use by the Quality Measure Engine, see section 3.3.

2.1. Measure Bundles

The Cypress Web Application requires one or more measure bundles to be loaded prior to use. Each measure bundle contains a set of CQMs and test patient data corresponding to a particular released version of the CQMs.

Measure bundles are downloaded separately from the Cypress software, a download link can be found on the Cypress test data Web page[1].

The separation of Cypress from the measure bundles allows new versions of MU CQMs to be deployed as they are released and would also allow entirely new measures to be used provided they authored using the same measure authoring tool[3] as the MU CQMs.

Section 3.1.1 contains additional information on the content of measure bundles.

2.2. User Interface

Cypress is an HTML5-based web application that leverages the jQuery JavaScript library with some additional plug-ins. The user interface has been tested on recent versions of Mozilla Firefox, Apple Safari, and Google Chrome.

Figure 2, Figure 3, and Figure 4 show key parts of the Cypress user interface.

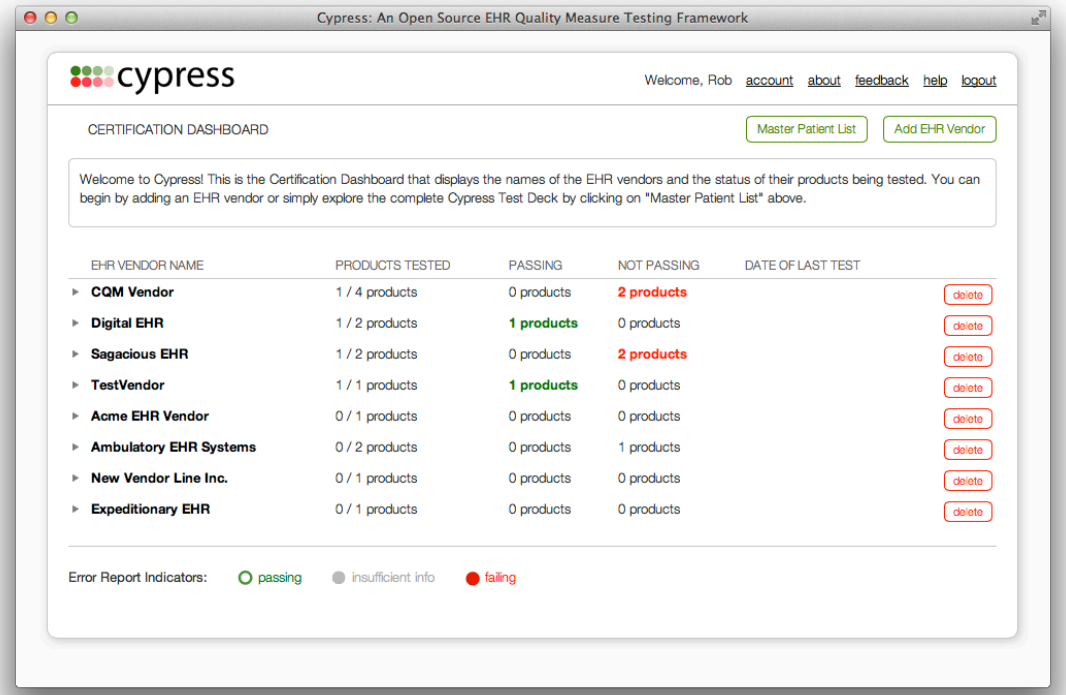

Figure 2: Certification dashboard showing the status of vendors and products under test.

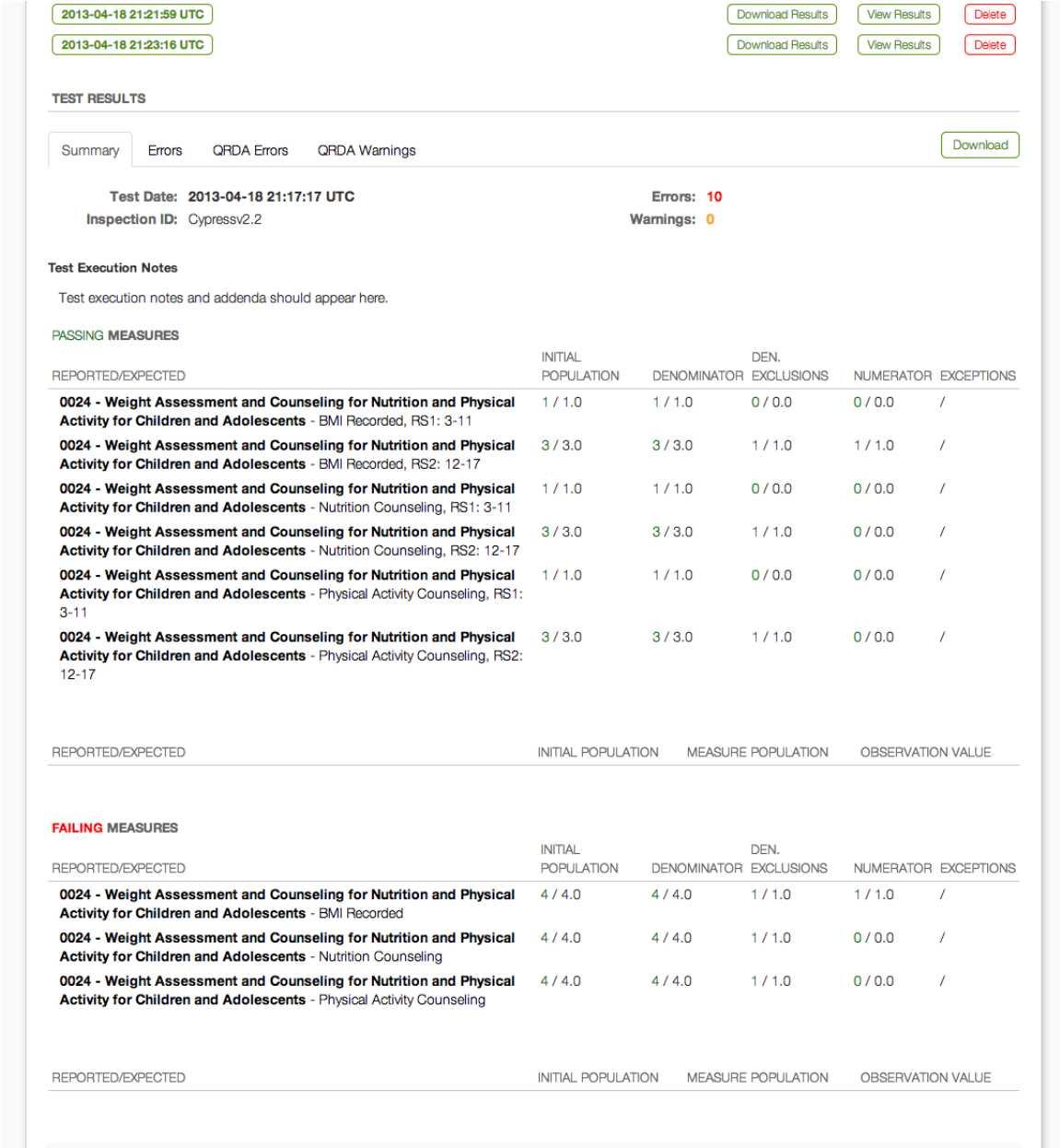

Figure 3: Product test dashboard showing passing and failing CQM calculation results

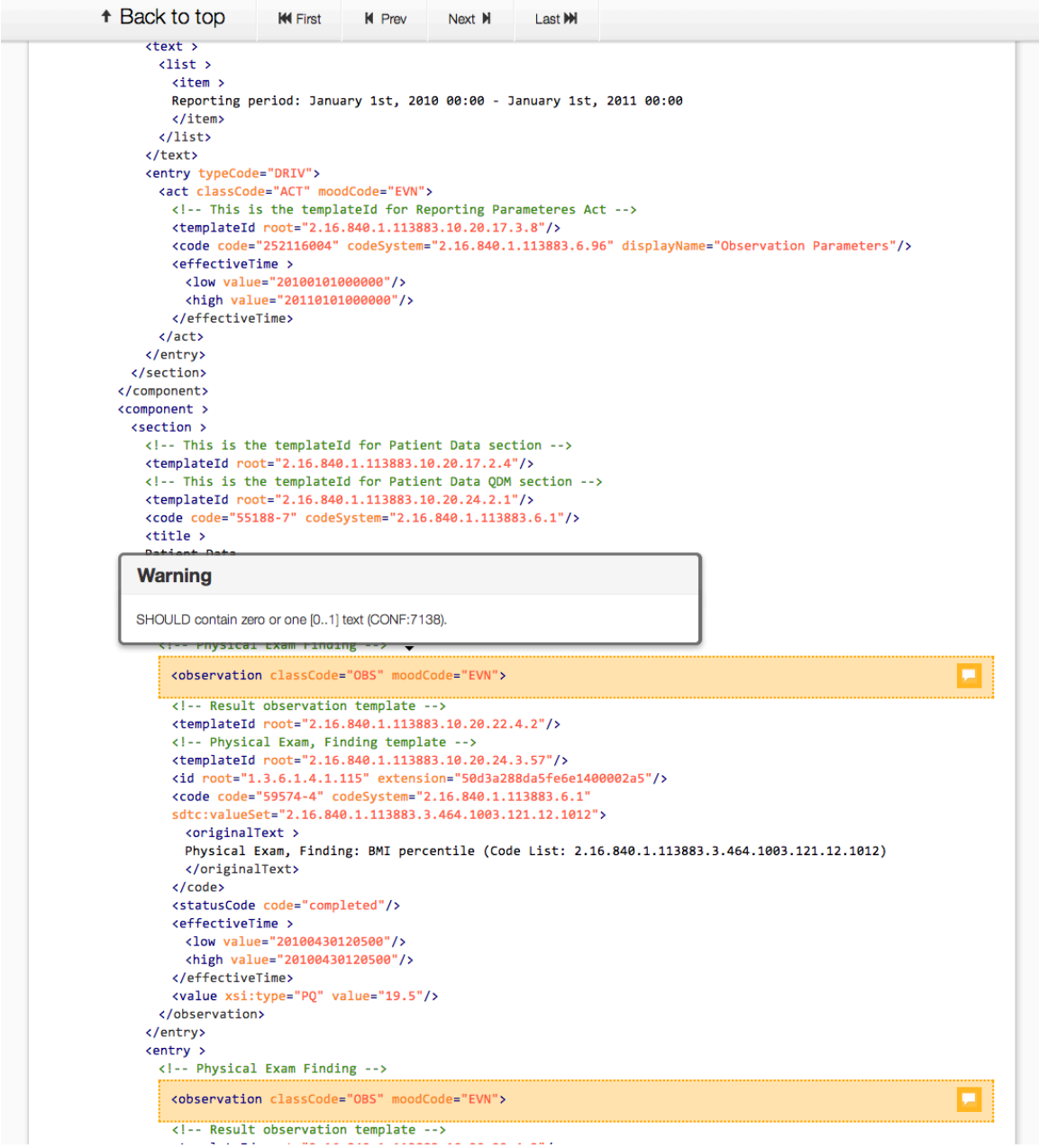

Figure 4: Test result detail showing QRDA XML validation issues.

3. Measure Calculation

Cypress' implementation of a CQM is generated directly from the CQM's HQMF definition, ensuring that CQM operators are implemented consistently across all CQMs. The XML-based definitions of the data elements and logic are translated programmatically into executable queries as shown in Figure 5.

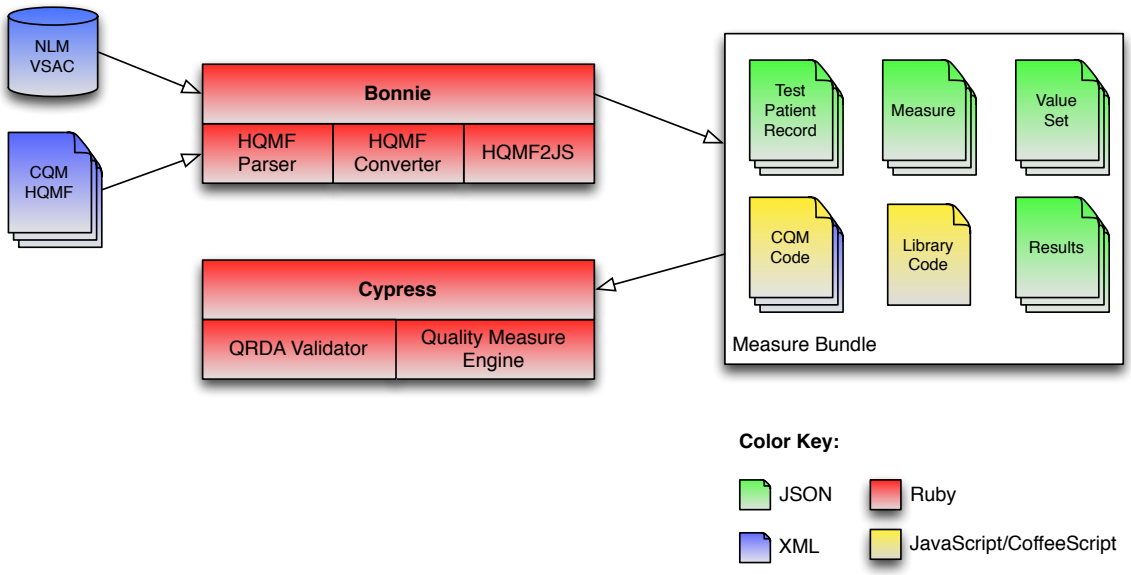

Figure 5: HQMF CQM Measure Calculation

The following subsections describe each of the components in Figure 5.

3.1. Bonnie

Bonnie[4] is a standalone application that provides the following capabilities:

1. Automated conversion of HQMF CQM logic into executable JavaScript.
2. Support for creation of patient test records that contain data of interest to CQMs.
3. Execution of CQM logic on test patients with reporting of trace data allowing the results of individual components of a measure to be viewed.
4. Measure debugging – JavaScript measure logic can be executed in a browser allowing full use of browser facilities for debugging measure logic.

Bonnie packages a set of CQMs, the required library code, and a corresponding set of test patient records into a measure bundle that can be loaded into Cypress.

3.1.1. Measure Bundle Contents

A measure bundle contains the following data:

- **Value Sets**
  Two files (one JSON, one XML) for each value set used by one of the included CQMs.
- **Measures**
  One JSON file for each CQM included in the Bundle. Each file contains metadata (title, description, etc) plus the generated code for that CQM.
- **Library Functions**
  Library code used by the generated JavaScript CQM code.
- **Patients**
  Test patient data, two files per patient: machine readable JSON and human readable HTML.
- **Results**
  Expected results for each CQM broken down by measure and test patient.
- **Sources**
  Source HQMF V1, converted HQMF V2, internal CQM representation in JSON.

The structure of the JSON patient data is defined by the record model in the Health Data Standards Library[5] and is illustrated below:

```
{
  "first": "John",
  "last": "Smith",
  "gender": "M",
  "birthdate": 1276869600,
  "encounters": [
    {
      "codes": {
        "CPT": [
          "99201"
        ]
      },
      "description": "Encounter, Performed: ...",
      "start_time": 1333274400,
      "end_time": 1333278000,
      "status_code": {
        "HL7 ActStatus": [
          "performed"
        ]
      },
      ...
    }
  ],
  "vital_signs": [...],
  ...
}
```

The above shows an extract of the test data record for “John Smith”, ellipses indicate where additional data has been omitted for brevity. Dates and times use the Unix seconds-since-epoch format, e.g. the birthdate corresponds to 6/18/2010. Following the patient demographics there are a number of subsections for

encounters, vital signs, procedures, etc. Each subsection contains a list of events, the above extract shows a single encounter coded using CPT 99201 with a start time, an end time, and a status of "performed". Additional properties will be present depending on the type of event, e.g. a test result would typically include a value.

3.2. HQMF Parser and HQMF Converter

MU stage 2 measures are defined using HQMF R1, an XML-based, CDA-derived format published as a draft standard for trial use (DSTU) by HL7. HQMF R1 is a complex format with a deeply nested structure that is both repetitive and difficult to comprehend. HQMF R2 was developed to address the complexity of R1 and Cypress bases its internal HQMF model on the R2 structure. Cypress parses HQMF R1 and then performs a two-pass conversion into its internal model using the HQMF parser and converter components of the Health Data Standards library[5].

3.3. HQMF2JS

The hqmf2js library[6] provides a translation from the internal HQMF model produced by the HQMF parser and converter to JavaScript suitable for use by the quality measure engine. The generated code relies on a number of custom-built JavaScript libraries (included as a part of hqmf2js) that implement functionality common to all measures. These libraries are described below.

3.3.1. Patient API

The patient API library[7] provides an object-oriented wrapper for the test patient data. The class structure is based on that of the "Green CDA for C32" schema with extensions for

- Additional data items required by MU, and
- Ease of use functions, e.g. to filter all of a patients encounters based on a value set

The patient API acts as a data access object (DAO) layer for the CQM generated code.

3.3.2. HQMF Library

The HQMF library implements a number of HL7 data types and all of the required QDM temporal, summary and subset functions. It also extends the patient API to provide conversions between patient API primitives and the HL7 data types. E.g. the patient API Event base class is extended to provide a method to yield an HL7 IVL_TS that captures the start and end timestamps of the event.

3.3.3. Logging

The logging library wraps patient API and HQMF library methods to generate a log of the execution of a CQM for each test patient record. The log details the inputs and outputs of each wrapped method and can be used to determine what data and population criteria a patient record satisfies and why. The log is very useful for tracking down discrepancies between expected and actual behavior of a particular CQM on a particular patient record.

Logs are stored in the database as part of the results cache, see section 3.4.3.

3.4. Quality Measure Engine

The quality measure engine manages execution of the generated JavaScript CQMs against a defined set of patient records. It utilizes the native map-reduce framework provided by MongoDB to execute measures. The quality measure engine is provided with a definition of the measure to execute including:

- The generated JavaScript map function
- The value sets used by the measure
- The Patient API, HQMF library and Logger libraries
- The type of measure (e.g., proportion or continuous variable?) - used to select a suitable reduce function (one per type of measure)

Measure calculation is processed in a background worker process using the Delayed Job Ruby gem.

3.4.1. Map-Reduce Introduction

Map-reduce is an algorithmic framework that can be used to calculate summary results over a large set of input data. It is widely used in "big data" problems, most notably by Google as the foundation of their Web search engine.

As its name suggests, map-reduce proceeds in two distinct steps, for the purposes of CQM calculation the first 'map' step takes as input a single patient record and maps it to a summary data format. The second 'reduce' step takes a set of summary data produced by the 'map' step and outputs a summary of that set. The format of the summary data differs depending on the type of measure. For patient-based proportion measures (denominator, numerator, etc) the format consists of a simple count of patients that meet the criteria for each population. Since the map step always works on one patient record at a time, the maximum count for any population will be 1[a]. The reduce function for proportion measures simply sums the counts for each population yielding a count of patients that met the criteria for each population. Figure 6 shows an illustration of this process for a proportion CQM.

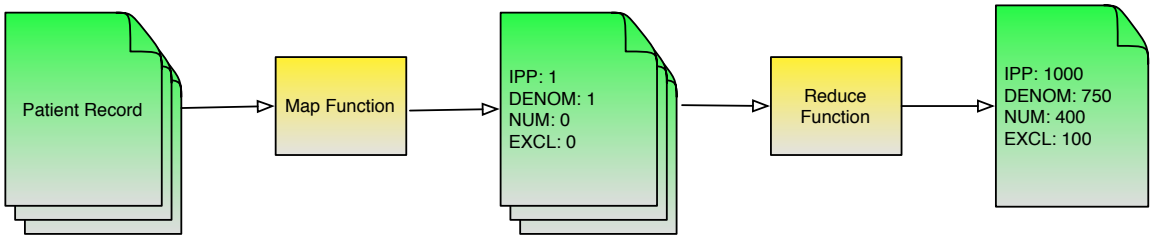

Figure 6: Illustration of Map-Reduce applied to a proportion CQM.

[a] Episode of care measures use the same format but the count for an individual patient may exceed 1 since one patient record may contain multiple matching episodes.

The reduce operation can be applied iteratively since its output format is the same as its input. E.g. you could evaluate the reduce function on groups of 100 results of the map function and then evaluate the reduce function a second time on the outputs of the first set of reduce functions to summarize over all groups.

Map-reduce has some important properties that make it suitable for large-scale calculations:

(i) Each map function evaluation is isolated and only requires access to a single input record to proceed. This allows many map functions to be run in parallel, either using separate threads within a process or distributing the work amongst a set of processes on a server farm.

(ii) Reduce functions can be applied iteratively allowing the work of reducing the output of map functions to be split amongst a set of workers and easily combined by a coordinator using a final reduce function execution.

The net result of (i) and (ii) is that native input data can easily be sharded[8] across a set of servers and a map-reduce calculation made across all shards with minimal coordination. Each shard can work in isolation and the results for each shard are easily combined

3.4.2. Cypress Use of MongoDB Map-Reduce Framework

For performance reasons, Cypress uses the native MongoDB map-reduce framework for the map step but supplies an identity function for the reduce step and a custom finalize step that annotates the output of the map step with additional measure information. The net result of this is that the annotated output of the map step (one entry per patient record per measure) is written to the patient_cache collection in MongoDB rather than being reduced in the traditional sense.

Cypress then performs its own separate reduce step using the MongoDB aggregation framework and writes the output of that (one entry per measure) to the query_cache collection in MongoDB.

3.4.3. Result Cache

As described above, for performance reasons, measure calculation results are cached after calculation in the MongoDB database. Results are cached in two MongoDB collections:

- **patient_cache**
  Stores the output of the map step, includes one entry for each combination of patient record, measure and effective date. Also includes the measure execution log for that combination, see section 3.3.3.
- **query_cache**
  Stores the output of the reduce step, includes one entry for each combination of measure and effective date.

4. QRDA Validation

Cypress performs validation of vendor-supplied QRDA files using the QRDA XML Schema plus a set of Schematron rules. The XML Schema tests the overall structural correctness of the files while the Schematron rules test that each data entry contains the required elements and that those elements obey the co-occurrence constraints defined by the QRDA specification.

5. Cypress and Open Source

Cypress is an open source project, licensed under the Apache 2[9] license. As an open source software project, Cypress endeavors to create a community dedicated to assuring the accuracy of automated clinical quality measure calculation.

All activities on the Cypress project are open. Not only does this include access to the underlying source code, but also planned features, forums, and all email discussions. Embracing an open source model allows us to build trust with a community of users, collaborators and vendors whose products are being certified using Cypress. With many eyes on all aspects of the Cypress project, we also hope to increase the reliability, accuracy, stability, and auditability of the software.

Examples of some of the successful open source software projects embracing these tenets include GNU/LINUX: a free open source operating system, Apache: the most popular web server used on the Internet, and Firefox: the second most popular web browser used on the Internet.

6. References

[1] Project Cypress: http://projectcypress.org/

[2] Project Cypress Code Repository: http://github.com/projectcypress/cypress

[3] CMS Measure Authoring Tool: https://www.emeasuretool.cms.gov/

[4] Bonnie Code Repository: https://github.com/pophealth/bonnie/

[5] Health Data Standards Code Repository: https://github.com/projectcypress/health-data-standards

[6] HQMF2JS: https://github.com/pophealth/hqmf2js

[7] Patient API: https://github.com/pophealth/patientapi

[8] Database Sharding: http://en.wikipedia.org/wiki/Shard_(database_architecture)

[9] Apache 2 License: http://www.apache.org/licenses/LICENSE-2.0.html

A. Acronyms and Terms

CDA – The HL7 Clinical Document Architecture is an XML-based markup standard intended to specify the encoding, structure and semantics of clinical documents for exchange.

CQM – A clinical quality measure. In the context of Cypress, this is actually an electronic specification of a CQM in the HL7 HQMF format.

EC2 – Amazon Elastic Compute Cloud allows users to rent computers on which to run their own computer applications. EC2 allows scalable deployment of applications by providing a web service through which a user can boot an Amazon Machine Image to create a virtual machine, which Amazon calls an "instance", containing any software desired.

EHR – Electronic Health Record. An EHR refers to an individual patient's health record in digital format. Electronic health record systems co-ordinate the storage and retrieval of individual records with the aid of computers.

FFRDC – Federally Funded Research and Development Center.

HHS – The United States Department of Health and Human Services. This is a Cabinet department of the United States government with the goal of protecting the health of all Americans and providing essential human services.

HL7 – Health Level Seven is an all-volunteer, not-for-profit organization involved in development of international healthcare standards.

HIT – Healthcare Information Technology or Health Informatics. This is the intersection of information science, computer science and health care. It deals with the resources, devices and methods required to optimize the acquisition, storage, retrieval and use of information in health and biomedicine.

HQMF – The Health Quality Measures Format is an HL7 XML standard for expressing Clinical Quality Measure logic, and the associated clinical codes.

jQuery – A cross-browser JavaScript library designed to simplify the client-side scripting of HTML. jQuery was released in January 2006 at BarCamp NYC. jQuery is free, open source software, dual-licensed under the MIT License and the GNU General Public License, Version 2. jQuery's syntax is designed to make it easier to navigate a document, select DOM elements, create animations, handle events, and develop Ajax applications. jQuery also provides capabilities for developers to create plugins on top of the JavaScript library. Additional information about the jQuery JavaScript framework is available at http://jquery.com

JSON – JavaScript Object Notation is a lightweight data-interchange format. It is easy for humans to read and write. It is easy for machines to parse and generate. It is based on a subset of the JavaScript Programming Language. JSON is a text format that is completely language independent but uses conventions that are familiar to programmers of the C-family of languages, which includes C, C++, C#, Java, JavaScript, Perl, Python, and others.

Map Reduce – A framework for processing large datasets on certain kinds of distributable problems using a large number of computers referred to as nodes. The framework supports two steps, a "*Map*" step where the master node takes the input, partitions it up into smaller sub-problems, and distributes those to worker nodes. A worker node may do this again in turn, leading to a multi-level tree structure. The worker node processes that smaller problem, and passes the answer back to its master node. There is also a "*Reduce*" step in this framework, where the master node then takes the answers to all the sub-problems and combines them in some way to get the output, the answer to the problem it was originally trying to solve. http://en.wikipedia.org/wiki/MapReduce

MITRE – The MITRE Corporation is a not-for-profit organization chartered to work in the public interest. As a national resource, MITRE applies expertise in systems engineering, information technology, operational concepts, and enterprise modernization to address their sponsors' critical needs.

NIST – The National Institute of Standards and Technology is a non-regulatory agency of the United States Department of Commerce. The institute's mission is to promote U.S. innovation and industrial competitiveness by advancing measurement science, standards, and technology in ways that enhance economic security and improve quality of life.

ONC – The Office of the National Coordinator for Healthcare Information Technology (ONC) is at the forefront of the administration's health IT efforts and is a resource to the entire health system to support the adoption of health information technology and the promotion of nationwide health information exchange to improve health care. ONC is organizationally located within the Office of the Secretary for the U.S. Department of Health and Human Services (HHS).

Open Source – Open source is a set of principles and practices that promote access to the design and production of goods and knowledge. This allows users to create software content through incremental individual effort or through collaboration.

Ruby – Ruby is a reflective, dynamic, object-oriented programming language. It combines syntax inspired by Perl with Smalltalk-like object-oriented features, and also shares some features with Python, Lisp, Dylan, and CLU. Ruby is a single-pass interpreted language.

Ruby on Rails – Ruby on Rails is a free web application framework. It aims to increase the speed and ease with which database-driven web sites can be created, and offers skeleton code frameworks (scaffolding) from the outset. Often shortened to Rails, or RoR. Ruby on Rails is an open source project written in the Ruby programming language.

REST – Representational State Transfer is a style of software architecture for distributed hypermedia systems such as the World Wide Web.

SOAP – SOAP refers to both Simple Object Access Protocol, and also lately Service Oriented Architecture Protocol. SOAP is a protocol for exchanging XML-based messages over computer networks.

Wiki – A type of computer software that allows users to easily create, edit and link web pages. Wikis are often used to create collaborative websites and power community websites.

XML – The Extensible Markup Language is a general-purpose markup language. It is classified as an extensible language because it allows its users to define their own tags. Its primary purpose is to facilitate the sharing of structured data across different information systems, particularly via the Internet.